Project 2 Report

I. Introduction

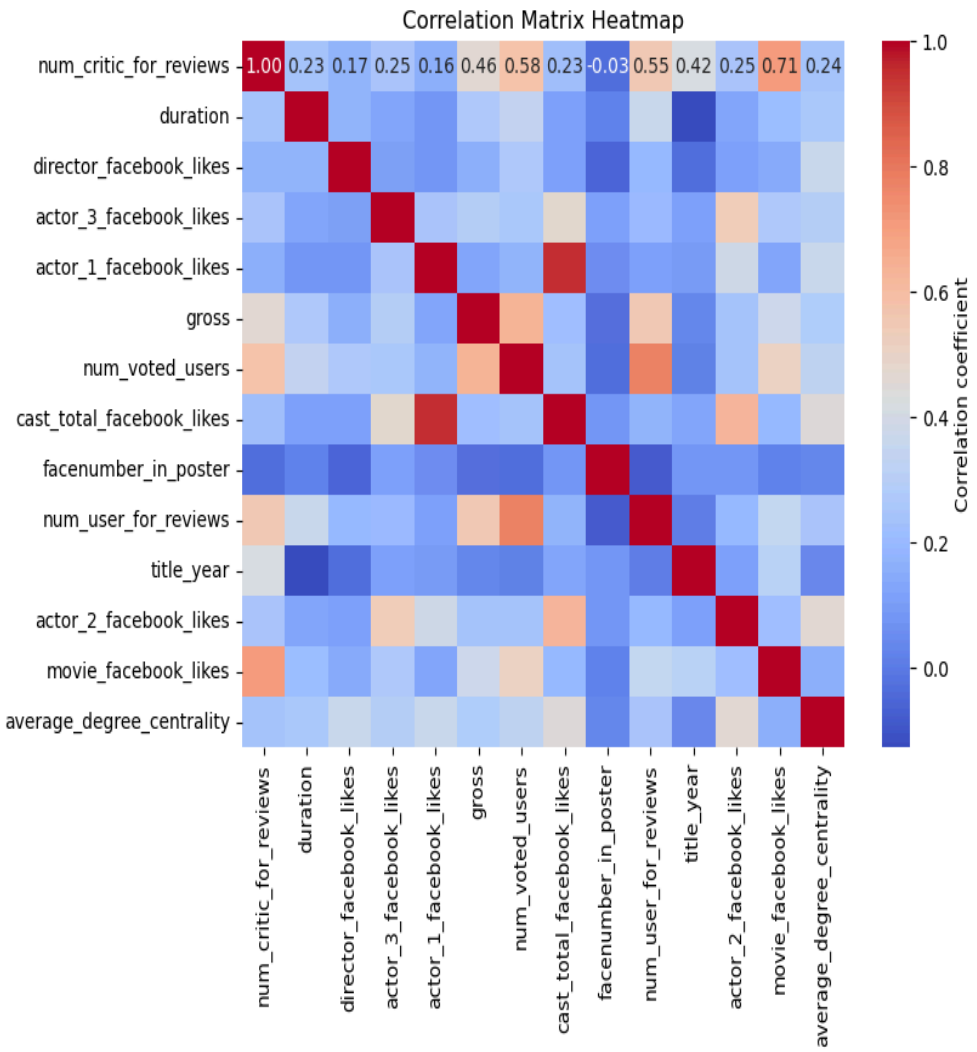
The task of this assignment is to implement different machine learning algorithms to predict imdb binned score of the movies on the test dataset. In this report, I am going to talk about all the models I have carefully implemented and analyzed, together with discussing the behavior of the models and identifying the problem of the data that make predicting challenging.

II. Methodology

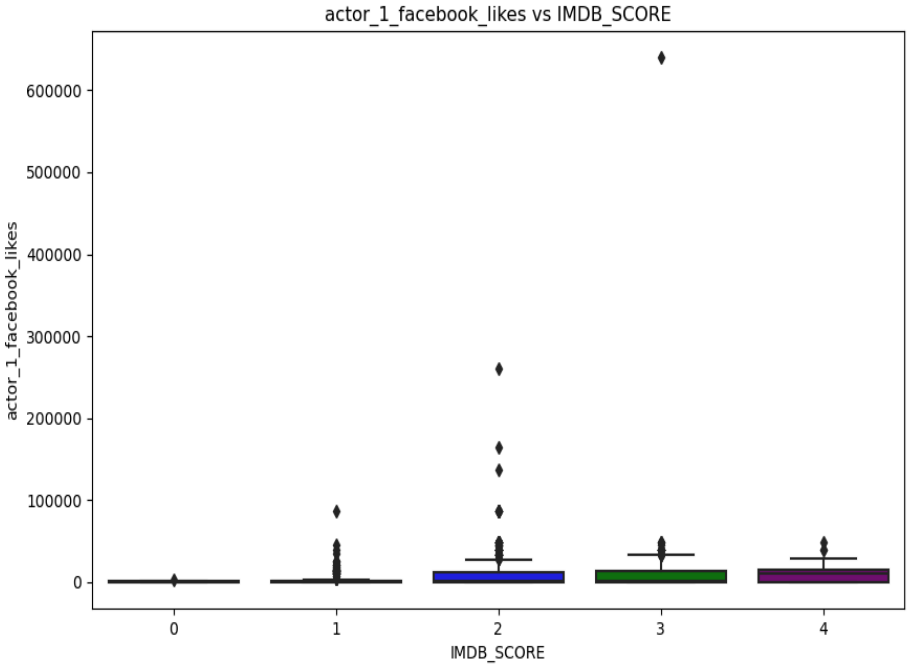
1.Data Pre-Processing and Features Engineering:

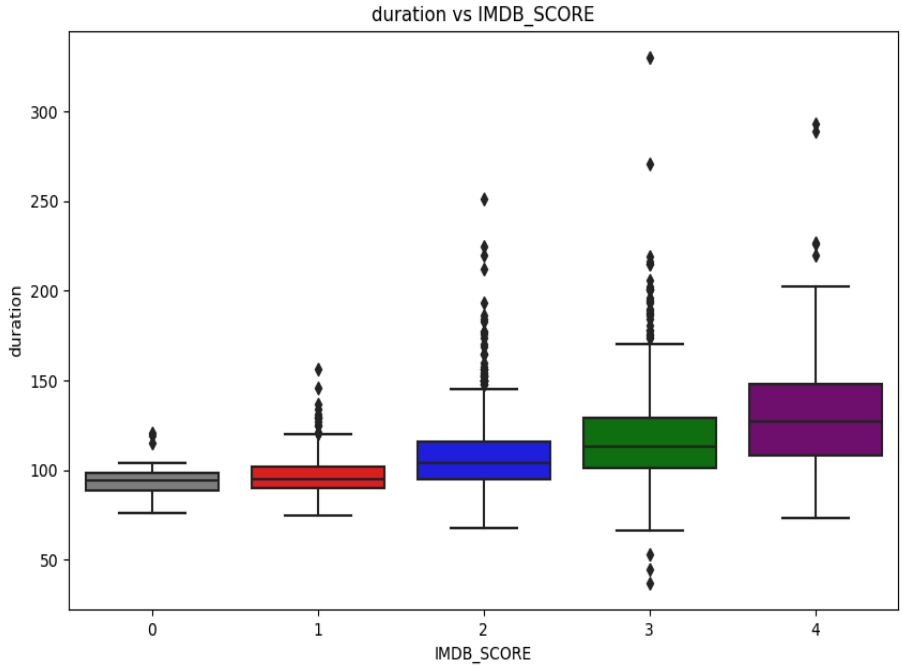
1.Data Visualization

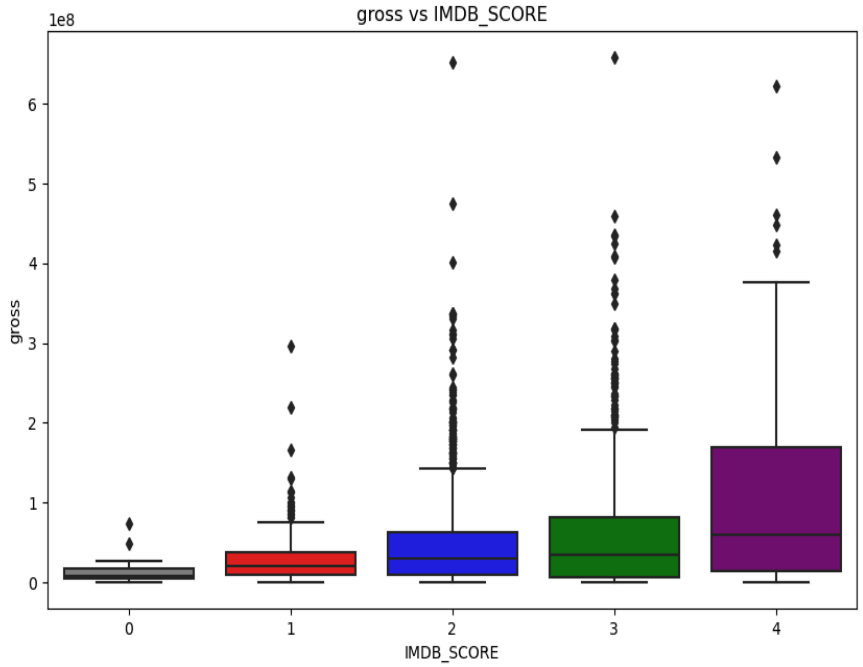
- We see that the dataset has a mixture of text type, continuous and categorical features. To make the data more simple, I will initially remove all text type and categorical data and focus on analyzing the correlation between continuous features.

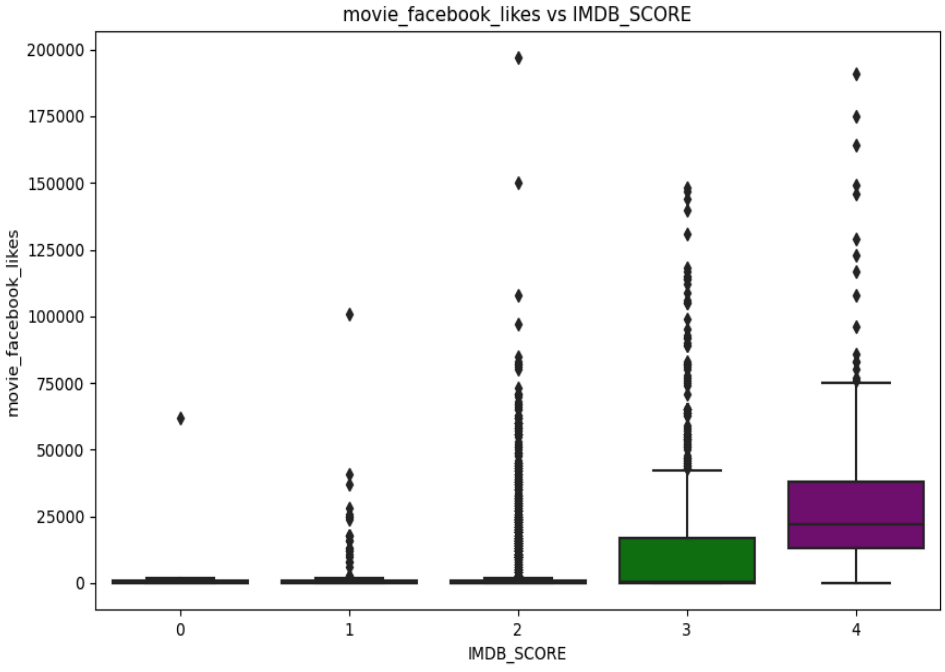


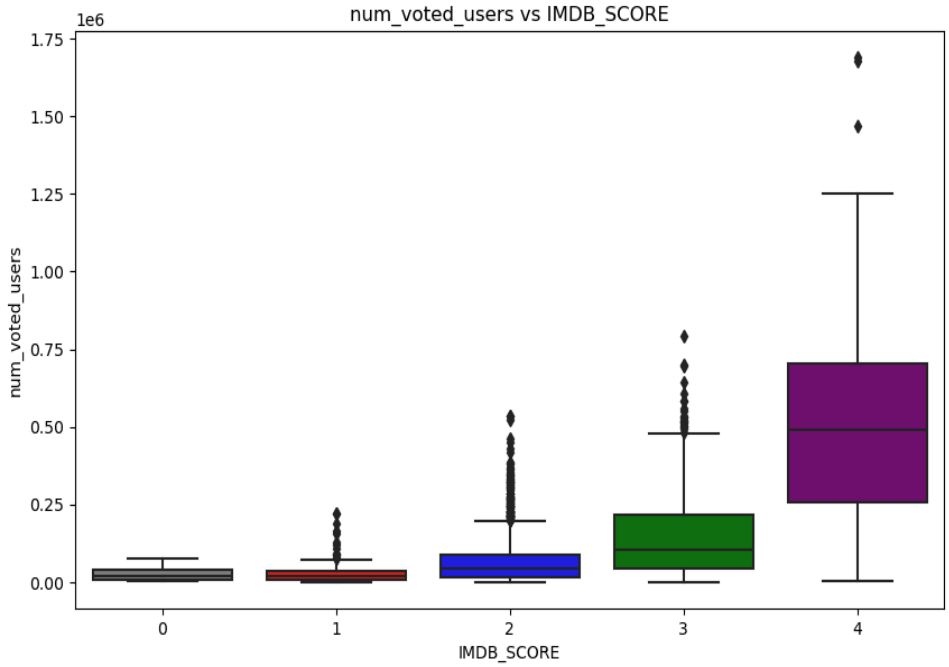
- On the correlation heatmap, I found that some features are moderately correlated with each other with correlation coefficient from 0.5-0.7, few are highly correlated such as movie_facebook_likes-num_critic_for_reviews, cast_total_facebook_likes-actor_1_facebook_likes, num_users_for_reviews-num_voted_users, the rest are either uncorrelated or have a insignificant correlation. This is not very good for classification process because the a lot of features are correlated in some way => we will need good features selection technique later.
- Next, we will look at at how those features and the label interact.

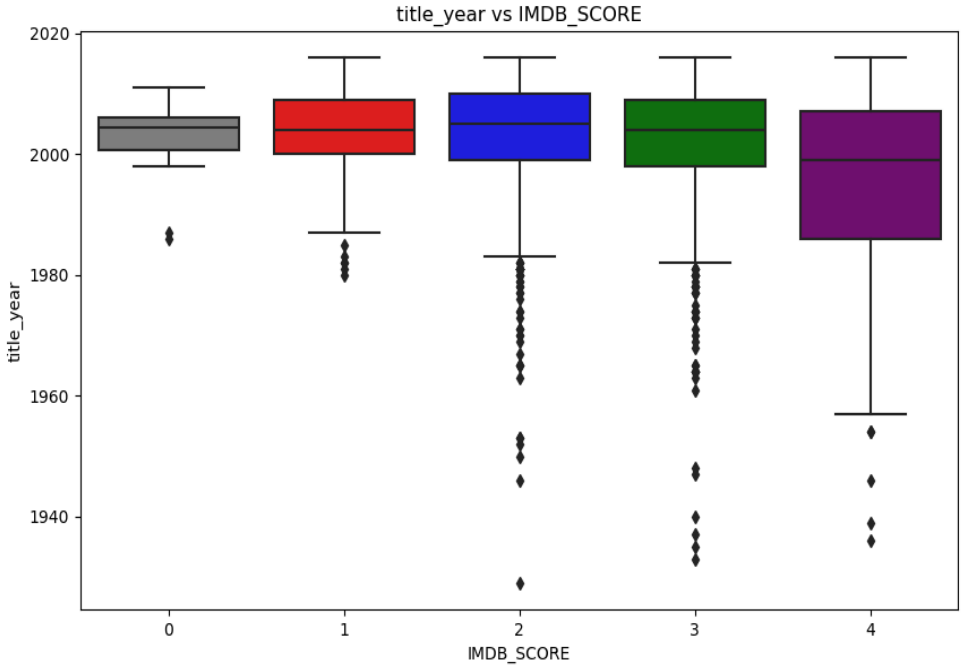

actor_1_facebook_likes vs IMDB_SCORE
actor_1_facebook_likes
600000
500000
400000
300000
200000
100000
0
0
1
2
3
4
IMDB_SCORE

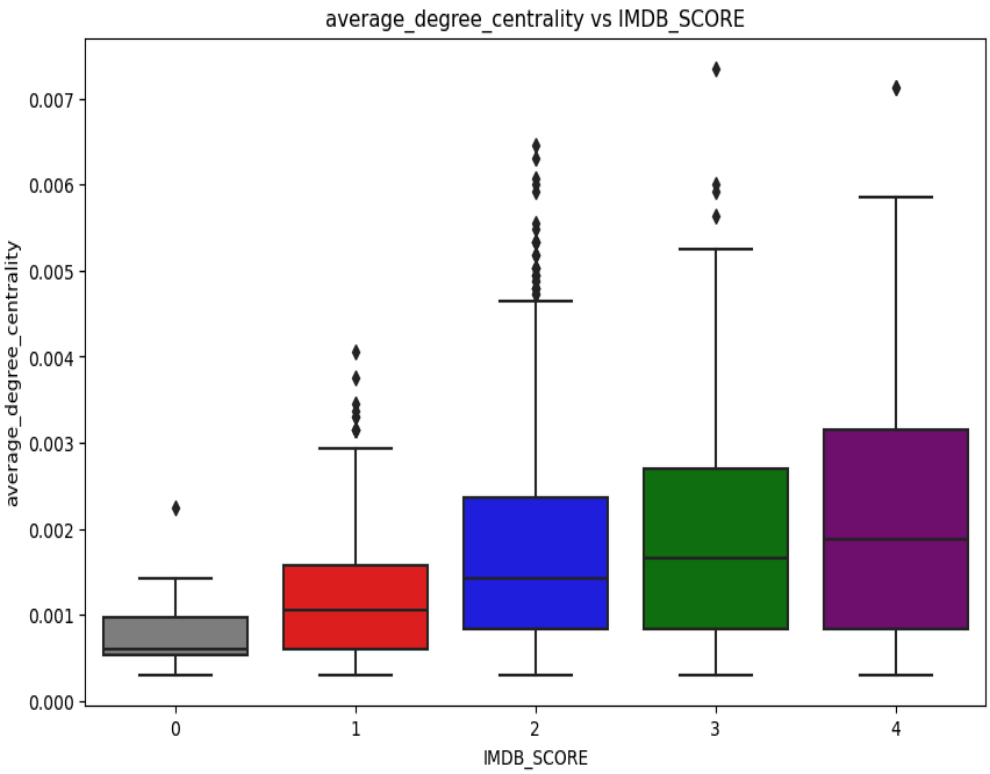

duration vs IMDB_SCORE
duration
300
250
200
150
100
50
0
1
2
3
4
IMDB_SCORE


gross vs IMDB_SCORE
1e8
gross
IMDB_SCORE
0
1
2
3
4
5
6


movie_facebook_likes vs IMDB_SCORE
movie_facebook_likes
IMDB_SCORE
0
25000
50000
75000
100000
125000
150000
175000
200000


num_voted_users vs IMDB_SCORE
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
num_voted_users
0
1
2
3
4
IMDB_SCORE


title_year vs IMDB_SCORE
2020
2000
1980
1960
1940
title_year
0
1
2
3
4
IMDB_SCORE



- We can really see a big issue with this data is class overlapping and outliers => Need to create more features, PCA, using complex models and outliers removal
- Let's have a look at class distribution of imdb_score_binned

Class	Count	Percentage
0	24	0.008
1	235	0.078
2	1839	0.612
3	777	0.259
4	129	0.043

We can see the dataset is very imbalance towards class 2 so we will need to use stratified sampling every time we want to sample train and test data for model evaluation.

2.Text Data and Categorical Data Processing

- For names of actors and directors, I will get rid of them because we already had their facebook likes, so those likes would be enough to represent actors and directors.
- For genres, plot_keywords and movie titles, I will keep the same processed genres and plot_keywords in the given files. After that, I will use PCA to reduce the dimension of genres and movie tiles being created for each of them separately. I do not use it for plot_keywords because we need the full dimension to capture enough useful information. The aim is achieving 90% plus of cumulative sum of reduced variance ratio.
- For language, country and content rating, I will use one hot encoder because these 3 are categorical data with < 50 unique values => probably a good choice. Then I will do PCA for each one just like above.

Feature	Cumulative Sum of Reduced Variance Ratio	New Dimension
genres	0.929	75
movie_title	0.933	75
language	0.869	10
country	0.923	10
content_rating	0.983	5

3. Outliers removal strategy

- I will remove outliers with the quantile method but the standard method with CQ1 = 0.25 and CQ3 = 0.75 will get rid of too many instances so we instead will do CQ1 = 0.005 and CQ3 = 0.995.

2. Models

1.Baseline

- I will use 1R model as our baseline model to compare our models to.

2.Decision Tree

- For the dataset with so many overlapping instances, I will use decision tree as one of the main algorithms for this assignment. However, I don’t

expect it to do well with the current dataset without features selection, normalization and outliers removal because the Decision Tree algorithm is unstable.
- For that reason, I will have to use grid search to find the appropriate parameters for the algorithm but I still don't expect much from this unstable algorithm.

3. Bagging Decision Tree

- To make up for that, I will first use feature selection with grid search to find the best number of features to be chosen, then use bagging algorithm with tuned decision tree to stabilize the algorithm.
- Then, I will also use grid search algorithm to find appropriate hyperparameters. I expect this algorithm to work a lot better than the decision tree algorithm alone.

4. AdaBoost Decision Tree

- I am going to try ada boosting algorithm with the tuned decision tree on feature selected data to see if it can help stabilize the decision tree like bagging with its unique focusing-on-hard-cases algorithm.

5. KNN

- Decision tree is an unstable algorithm so I will try to use another stable algorithm to compare their performance.
- For KNN, I will normalize the data to ease up the computation cost, then I will grid search with feature selection to find the best features subset. I will also use grid search to find appropriate hyperparameters for knn.
- For the latter algorithms, I also used standardized feature-selected datasets.

6. Random Forest

- I intend to use it to be the meta classifier for stacking cause I was impressed on how powerful it sounds in the lecture, it literally has no weaknesses except for interpretability.

7. Stacking

- After making different models, I will stack bagging decision trees, adaboost decision trees and knn, then use random forest as the meta classifier to see if the prediction improves.

8. Repeat with outliers removal

- I will do feature selection on this new data again, do grid search with bagging decision trees, knn, adaboost and random forest. Then stack them to predict and see if outliers removal helps improve the performance.

III. Results

Model	Cross Validation	Performance
1R	0.63	0.60
Decision Tree	0.58	0.61
Tuned Decision Tree	0.64	0.62
Bagging Decision Tree	0.7	0.66
AdaBoost	0.68	0.66
KNN	0.68	0.678
Random Forest	0.69	None
Stacking	0.69	0.69
Stacking with outliers removal	0.71	0.66

IV. Discussion and Critical Analysis

According to the result, I have come up with a lot of discussions and analysis:

- Decision Tree does not work well on high-dimensional. In high-dimensional spaces, data points are typically far apart from each other. This sparsity makes it difficult for decision trees to make splits that effectively segment the data into homogeneous or near-homogeneous groups. So the grid search figured out that by limiting the depth of the tree to 10, we can prevent overfitting and make the data less noise-sensitive, but a single tree is not stable enough to perform well on such a noisy dataset.
- Bagging really helps stabilize the Decision Tree a lot better. Because it is effective over noisy dataset as outliers may vanish during the process.
- AdaBoost also help improve the performance of Decision Tree as it focuses on the hard-to-classify instance => very nice for this overlapping dataset, it also handle overfitting by building shallow trees and combine them and it is less concerned about the data distribution => works well on this imbalanced dataset. However, it did not do any

better than Bagging but the computation is more expensive and more prone to overfitting.
- KNN did a great job as it is much more stable than Decision Tree, together with normalization, feature selection and hyperparameter tuning.
- Random Forest works really well as the meta classifier as it is robust to overfitting and can handle imbalanced data well.
- Stacking different models that work fairly well individually actually improves the performance overall. However, I am not sure if stacking complex ensemble classifiers is a really good choice but for this assignment it improved my performance.
- Normalization and Feature Selection plays an important role as expected as they make computation easier, improve KNN and get rid of correlated features as there are a lot of them when we plot the heatmap.
- Hyperparameter tuning is an incredibly powerful tool to use as it optimizes the model by choosing the best hyperparameters set. However, if the cross validation scores of hyperparameters sets are close, then its effectiveness will drop.
- Removing outliers surprisingly downgrades the performance of my stacking model. It is probably because removing outliers means sacrificing training data => cross validation score increases as there is less data to evaluate and no more outliers but performance decreases as there is not enough training data.

V. Conclusion

Overall, I came to the conclusion that the algorithm we use can only take us so far if our data is not pre-processed properly. I have learned that doing grid search on SelectKBest is not very reliable as it can produce different results so maybe it is sometimes better to do hyperparameter tuning manually. In my opinion, the main problem of this data is that the training dataset is too small so we cannot get enough data to train and to produce accurate predictions on such noisy and overlapping data.